Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение

высшего образования

Московский технический университет связи и информатики

(МТУСИ)

Кафедра «Информационная безопасность»

ЛАБОРАТОРНАЯ РАБОТА №7

по дисциплине

«Программирование в системах информационной безопасности»

на тему

«Динамическое выделение памяти»

Вариант №25

Выполнил:

студент группы БСУ1801 Убушуев Б.С

Проверил:

старший преподаватель кафедры ИБ

Барков В. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2021

# **Цель**

Овладеть навыками выделения динамической памяти в языке программирования C.

# **Задание 1**

Написать функцию, которая создаёт в динамической памяти целое число, осуществляет ввод, выводит на экран адрес переменной и её значение, увеличивает значение на единицу и снова выводит адрес и значение на экран. Перед выходом из функции необходимо освободить ранее выделенную память.

Прототип функции void incrementHeapVariable().

Формат вывода: адрес значение адрес значение.

Листинг 1 – программный код файла task1.c

#include <stdio.h>

void incrementHEapVariable(void)

{

int \*p = (int \*)malloc(sizeof(int));

printf("Введите число ");

scanf\_s("%d", p);

printf("Адрес переменной: %p Значение: %d.\n", p, \*p);

++\*p;

printf("После увеличения на 1 - %d\n", \*p);

free(p);

}

# **Задание 2**

Написать функцию, которая создаёт в динамической памяти из N элементов (число N вводится пользователем с клавиатуры), выводиn n на экран и с помощью разработанных функций (практикум №4, задания 2-4) осуществляет ввод данных, вывод массива на экран, сортировку и повторный вывод отсортированного массива на экран. Перед выходом из функции необходимо освободить выделенную ранее память.

Прототип функции void sortHeapArray(int \*arr, int n).

Формат вывода (первая строка – количество элементов массива):

5

5 1 9 7 8

1 5 7 8 9

Листинг 2 – программный код файла task2.c

#include <stdio.h>

void sortHeapArray(int \*arr, int n)

{

readArray(arr, n);

printf("========== \n", n);

printf("%d \n", n);

writeArray(arr, n);

sort(arr, n);

writeArray(arr, n);

free(arr);

}

# **Задание 3**

Написать функцию, удаления подстроки в строке. Функция принимает исходную строку и подстроку для поиска. С помощью функции find найти адрес начала подстроки и затем удалить эту подстроку. Функция выделяет необходимое количество памяти в области динамической памяти, копирует необходимые части строки и возвращает указатель на вновь выделенную память. Обращение к символам строки после нулевого символа недопустимо.

Прототип функции char \*delete(char \*str, const char \*substr).

Функция не должна использовать функции консольного ввода-вывода.

Листинг 3 – программный код файла task3.c

char \*delete(char \*str, const char \*substr)

{

unsigned int strLen = strlen(str);

unsigned int substrlen = strlen(substr);

char \*point = find(str, substr);

char \*newstr = (char\*)malloc((strLen - substrlen+1), sizeof(char));

char \*curDst = newstr;

for (const char \*curSrc = str;

\*curSrc;

++curSrc, ++curDst)

{

if (curSrc == point)

{ curSrc += substrlen; }

\*curDst = \*curSrc;

}

\*curDst = 0;

return newstr;

}

# **Задание 4**

Написать функцию копирования строк. Функция принимает исходную строку, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое исходной строки и возвращает указатель на вновь выделенную память

Прототип функции char \*strcpy(const char \*src).

Функция не должна использовать функции консольного ввода-вывода.

Листинг 4 – программный код файла task4.c

#include <malloc.h>

char \*strcpy(const char \*src)

{

int len = strlen(src);

char \*dest = (char\*)malloc(len, sizeof(char));

char \*curDst = dest;

for (const char \*curSrc = src;

\*curSrc;

++curSrc, ++curDst)

{

\*curDst = \*curSrc;

}

\*curDst = 0;

return dest;

}

# **Задание 5**

Написать функцию конкатенации строк. Функция принимает исходные строки, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое строк и возвращает указатель на вновь выделенную память

Прототип функции char \*strcat(const char \*first, const char \*second).

Функция не должна использовать функции консольного ввода-вывода.

Листинг 5 – программный код файла task5.c

char \*strcat (const char \*first, const char \*second)

{

unsigned int strLen = strlen(first);

unsigned int substrlen = strlen(second);

char \*newstr = (char\*)malloc(strLen + substrlen + 1, sizeof(char));

char \*curDst = newstr;

for (const char \*curSrc = first;

\*curSrc;

++curSrc, ++curDst)

{

\*curDst = \*curSrc;

}

for (const char \*curSrc = second;

\*curSrc;

++curSrc, ++curDst)

{

\*curDst = \*curSrc;

}

\*curDst = 0;

return newstr;

}

# **Задание 6**

Написать функцию вставки подстроки в строку. Функция принимает исходнную строку, строку для вставки, индекс, определяющий место вставки, выделяет необходимое количество памяти в области динамической памяти, копирует реузльтат и возвращает указатель на вновь выделеную память.

Прототип функции char \*insert(const char \*src, const char \*str, int index).

Функция не должна использовать функции консольного ввода-вывода.

Листинг 6 – программный код файла task6.c

char \*insert(const char \*src, const char \*str, int index)

{

unsigned int strLen = strlen(str);

unsigned int srcLen = strlen(src);

char \*newstr = (char\*)malloc((strLen + srcLen + 1 ), sizeof(char));

char \*curStr = newstr;

char \*p = src + index;

for (const char \*curSrc = src;

\*curSrc;

++curSrc, ++curStr)

{

if (curSrc == p)

{

for (const char \*curSrc = str;

\*curSrc;

++curSrc, ++curStr)

{

\*curStr = \*curSrc;

}

}

\*curStr = \*curSrc;

}

\*curStr = 0;

return newstr;

}

# **Задание 7**

Модифицировать приложение из практикума №6, заменив статические массивы динамическими. Размеры всех массивов определяются количеством записей (количество семестров равно количеству сессий, включая текущую, количество предметов равно количеству сданных на текущий момент экзаменов и зачётов, все строковые массивы должны иметь длинну, достаточную для хранения всех символов строки, включая нулевой символ).

Листинг 7 – программный код файла l4func.c

#include <stdio.h>

//task2

void writeArray(int \*arr, int n)

{

for (int i = 0; i < n; i++)

{

printf("%d ", arr[i]);

}

printf("\n");

}

//task3

void readArray(int \*arr, int n)

{

for (int i = 0; i < n; i++)

{

scanf\_s("%d", &arr[i]);

}

}

//task4

void mergeSort(int \*a, int l, int r)

{

if (l == r) return;

int mid = (l + r) / 2;

mergeSort(a, l, mid);

mergeSort(a, mid + 1, r);

int i = l;

int j = mid + 1;

int \*tmp = (int\*)malloc(r \* sizeof(int));

for (int step = 0; step < r - l + 1; step++)

{

if ((j > r) || ((i <= mid) && (a[i] < a[j])))

{

tmp[step] = a[i];

i++;

}

else

{

tmp[step] = a[j];

j++;

}

}

for (int step = 0; step < r - l + 1; step++)

a[l + step] = tmp[step];

}

void sort(int \*arr, int n) {

mergeSort(arr, 0, n - 1);

}

Листинг 8 – программный код файла l5func.c

int strlen(const char \*str)

{

int i = 0;

while (\*str)

{

++str;

++i;

}

return i;

}

char \*find(const char\* str, const char\* substr)

{

unsigned int i = 0;

unsigned int j, k;

while (str[i] != 0)

{

for (j = i, k = 0; substr[k] != 0 && str[j] == substr[k]; j++, k++)

{

if (substr[k + 1] == 0)

{

return (str + i);

}

}

++i;

}

return 0;

}

Листинг 9 – программный код файла l7.c

#include <stdio.h>

#include <stdbool.h>

#include <locale.h>

#include "Header.h"

void main(void)

{

setlocale(LC\_ALL, "Rus");

\_Bool close = true;

while (close)

{

char str1[] = "visual studio";

char substr[] = "al";

char str3[] = "hrhrhr";

unsigned int len1 = 0, destlen = 0, index = 0;

\_Bool err;

char \*point , dest[100];

printf("1. Задание 1\n");

printf("2. Задание 2\n");

printf("3. Задание 3\n");

printf("4. Задание 4\n");

printf("5. Задание 5\n");

printf("6. Задание 6\n");

printf("7. Выход \n");

switch (getch())

{

case '1':

system("cls");

len1 = strlen(str1);

printf("Строка - %s. Длина строки - %d \n", str1, len1);

break;

case '2':

system("cls");

point = find(str1, substr);

point == 0 ? printf("Строка - %s. Подстрока - %s. \nПодстрока не входит в строку \n", str1, substr) : printf("Строка - %s. Подстрока - %s. \nАдрес вхождения - %p \n", str1, substr, point);

break;

case '3':

system("cls");

printf("Строка - %s. Подстрока - %s \n", str1, substr);

delete(str1, substr);

printf("Строка после удаления подстроки - %s \n", str1);

break;

case '4':

system("cls");

destlen = strlen(dest);

strcpy(dest, destlen, str1);

printf("Строка - %s. После копирования - %s \n", str1, dest);

break;

case '5':

system("cls");

destlen = strlen(dest);

strcat(dest, destlen, str1, str3);

printf("Строка - %s. Вторая строка - %s. \nРезультат конкатенации - %s \n", str1, str3, dest);

break;

case '6':

system("cls");

destlen = strlen(dest);

printf("Введите индекс для вставки ");

scanf\_s("%d", &index);

err = insert(str1, str3, index, dest, destlen);

err ? printf("Строка - %s. Вторая строка - %s. \nРезультат вставки - %s \n", str1, str3, dest) : printf("Ошибка \n");

break;

case '7':

close = false;

break;

}

}

}

Листинг 10 – программный код файла Header.h

void incrementHEapVariable(void);

void writeArray(int \*arr, int n);

void readArray(int \*arr, int n);

void sort(int \*arr, int n);

void sortHeapArray(int \*arr, int n);

void quickSort(int \*arr, int left, int right);

char \*find(const char\* str, const char\* substr);

int strlen(const char \*str);

char \*delete(char \*str, const char \*substr);

char \*strcpy(const char \*src);

char \*strcat(const char \*first, const char \*second);

char \*insert(const char \*src, const char \*str, int index);



Рисунок 1 – Результат работы с программой (Задание 1)

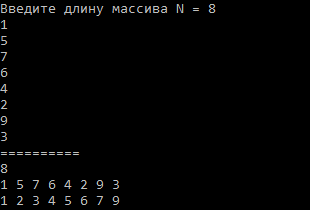


Рисунок 2 – Результат работы с программой (Задание 2)



Рисунок 3 – Результат работы с программой (Задание 3)



Рисунок 4 – Результат работы с программой (Задание 4)



Рисунок 5 – Результат работы с программой (Задание 5)



Рисунок 6 – Результат работы с программой (Задание 6)

**Вывод**

В данной работе были разобрана работа с динамической памятью в языке программирования С.